利用ε-贪婪学习和用户行为反馈的搜索引擎网页排序算法 *

张春玲 [1]，姜成晶 [2]

(1. 潍坊科技学院 中印计算机软件学院, 山东 寿光 262700; 2. 韩国圆光大学 计算机软件工程学院, 韩国 54538)

摘 要：搜索引擎是根据用户查询对相关网页进行排序。为了提高网页排序的准确性，提出一种基于ε-贪婪学习和用户点击行为的网页排序算法。首先，根据用户查询，通过轮盘赌策略向用户推荐相关网页列表；然后，根据用户点击网页的行为进行ε-贪婪学习，计算得到排序系统中的强化信号，通过奖励和惩罚机制为每个网页计算相关性程度值；最后，根据相关性程度对网页进行重新排序。随着用户反馈的信息越来越多，相关网页会排列在列表的最高等级上。实验结果表明，提出的算法能够准确推荐出相关网页，在 P@n、NDCG 和 MAP 性能指标上都获得了较优的性能。

关键词：搜索引擎；网页排序；ε-贪婪学习；用户行为

中图分类号：TP311　　doi: 10.3969/j.issn.1001-3695.2018.02.0082

Page ranking algorithm based on ε-greedy and user behavior in search engine

Zhang Chunling[1], Sun-kyoung Kang[2]

(1. *Sino-India Computer Software Institute*, *Weifang University of Science & Technology*, *Shouguang Shandong* 262700, *China*; 2. *Department of Computer-Software Engineering*, *Wonkwang University*, *Iksan* 54538, *South Korea*)

Abstract: Search engine is a tool that ranks related Web pages based on user queries. In order to improve the accuracy of Web page ranking, this paper proposed a Web page ranking method based on ε-greedy learning and user click behavior. Firstly, it recommend to the user a list of related Web pages by the roulette strategy according to the user query. Then, it performed ε--greedy learning based on the behavior of the user clicking on the Web page, and calculated the fortified signal in the ranking system. After that, it calculated the relevancy degree value for each Web page through reward and punishment mechanisms. Finally, it reordered the network according to the degree of relevance. As more and more information is fed back by users, related Web pages would be ranked at the highest level of the list. The experimental results show that the proposed method can accurately recommend the relevant Web pages and obtain better performance on P@n, NDCG and MAP performance indexes.

Key words: search engine; page ranking; ε-greedy learning; user behavior

0　引言

网页搜索引擎为网络用户搜索信息提供了很大的便利。搜索过程从用户提供查询开始，通过搜索算法为用户提供一个结果列表。搜索引擎面临的主要挑战就是对网页进行正确排序。排序算法用来对用户查询结果进行排序，即根据其相关性从高到低进行降序排列[1]。

目前提出的网页排序方法根据内容和连接性分为两大类[2]。基于内容的方法根据网页内容进行推荐，而且网页内容由其创建者决定。基于连接的方法则分为独立于查询和依赖于查询两种。这种方法存在的问题是主流页面会更流行，用户不会看到其他新的相关页面[3]。组合方法则同时注重内容和连接性两方面，可以减少这些问题的出现。

一些情况下，在输入类似查询时，用户的意图却不一定相同，可以利用用户的兴趣来解决查询的模糊性[4]，所以基于用户反馈的算法可以提供很好的结果。文献[5]研究表明，在用户点击网页搜索结果的频率分布方面，用户更倾向于点击排在第一级的网页，网页相对的点击数量随着排名降低而下降，其中用户点击第二、第三和第四级网页的概率约 60%、50%和 30%。这表明，即使排在搜索结果中的高级别网页是无关的，用户也会点击。因此，用户点击网页的行为，本质上是有干扰的。即用户经常点击低质量的结果，仅有 82%的被点击的网页与查询主题相关[6]。但尽管如此，用户的点击可成为涉及网页排序的有用知识。另一方面，机器学习作为一种强有力的工具，可以创造出比基本方法更好的排序结果，而如何在训练后产生一个适合新查询排序的模型，则是一个研究重点。

本文提出了一种基于 ε-贪婪学习和用户行为反馈的排序算法，称为 GLUB-Rank。为了了解用户的兴趣，用户需要参与

收稿日期：2018-02-05；**修回日期：**2018-03-12　　**基金项目：**国家自然科学基金资助项目（61402067）

作者简介：张春玲（1980-），女，山东寿光人，讲师，硕士，主要研究方向为网页排序算法（zhangchunling2017@126.com）；姜成晶（1976-），女（朝鲜族），韩国群山人，副教授，博士，主要研究方向为智能算法等.

到学习过程中。ε-贪婪学习是强化学习的一种，可以在不需要基础知识的情况下，获得与环境相关的知识，选择网页向用户显示，并为下一个类似的查询提供适当的排序。

1 相关研究

近年来，基于学习的排序已经成为信息检索领域的研究热点。基于学习的排序方法分为逐点法、成对法和列表法三个大类。

逐点法是学习方法中最简单的方法，这种方法的思想是将每个“查询-网页”对映射到相关的数字值。线性回归（linear regression）是一种基于统计学的逐点法，其旨在学习一个线性排序函数，其中将特征向量映射到实际值。文献[7]提出了一种基于概率的算法：McRank，包括多重分类的增量树算法和有序的多重分类，该方法使用组合排序来最小化错误排列对的数量。文献[8]提出的 FP-Rank 算法结合了 PageRank 和 TF-IDF 特征，并利用了用户反馈。此方法基于三个组件进行自适应调整：连接性，内容和用户行为。其中代理的目标是最大化高质量页面上的点击次数。文献[9]提出的 DistanceRank 算法是基于学习来计算页面对数距离的算法，其目的是尽量减少总损失，距离计算过程持续到它收敛到一个常数值。

在成对法中，学习的样本是网页对，而学习问题是根据分类器制定的。这种方法从列表中提取网页对，每个网页对都有一个标签，其考虑到了两个网页之间的部分相关性。然后，用标号数据对模型进行分类，并在排序中进行模型训练。文献[10]提出了基于学习的 RankBoost 算法，该算法的作用类似于基于聚类网页的 AdaBoost 算法，唯一的区别是 RankBoost 是在一对网页上定义的。文献[11]提出了一种基于学习的成对算法，称为 RankSVM。其用 SVM 对网页的二进制分类问题进行排序，点击排序网页的概率直接关系到查询—网页的相关性。

与其他两种方法相比，列表法效率更高。假定所以网页对或网页点具有特定的特征，特征选择在信息检索中是有偏差的，且依赖大量的查询。存在一种名为 ListNet 的列表式学习方法[12]，它使用 k 个高概率的排序结果来优化损失函数，采用神经网络作为模型，用梯度下降法代替优化算法。但是 ListNet 算法具有较高的时间复杂度。文献[13]提出了 BoltzRank 排序方法，它使用条件概率分布，将网页分级到用户的查询，其思想是为网页的排列和性能评价预测定义一个概率分布。文献[14]提出了一种基于 PSO 的 SwarmRank 排名算法，该算法试图学习许多排序函数的线性组合，其目标是优化排序的平均准确率性能指标。文献[15]提出了一种基于学习自动机的排序算法 LRUF，其利用了用户反馈信息。LRUF 算法根据排序列表中的每个网页的位置进行排序，并对更新的评分进行排序。在此方法中，当网页被选择的概率很低时，会被删除，并被其他网页所取代，从而减少了“富者愈富”的效应。因为这种算法需要计算每个步骤中所有网页的概率，所以算法的计算复杂度较高。

2 ε-贪婪学习

强化学习是一种通过奖励或惩罚来训练代理执行某一行为的方法，而不需要指定该行为对代理的作用。其目标是找到最优策略，以使所有状态的预期值最大化。强化学习的框架如图 1 所示。

强化学习有两种主要策略：一种是利用演化算法在行为空间中寻找行为，从而达到目标；另一种是使用统计方法和动态规划。

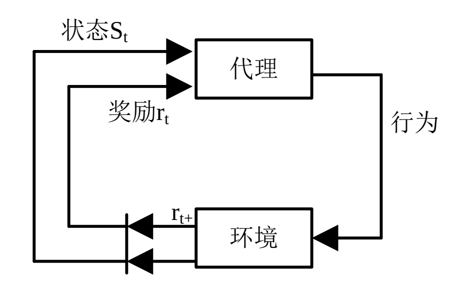


图 1　强化学习的框架

ε-贪婪学习是强化学习方法之一。本文利用 ε-贪婪方法来学习用户点击反馈，以此给出推荐模型。在这种方法中，代理能够在没有完美环境模型下选择最优策略。环境进入下一个状态，并将强化信号提供给代理。此时会产生一个行为值，这个值是对未来行为状态所受奖励的估计。

令 $Q_k(a)$ 表示在 k 时刻的行为 a 的估计值，a_k^* 表示在 k 时刻选定的行为。ε-贪婪方法通过使用概率 1-ε 来选定一个具有最高估计值的行为，即 $a_k^* = \arg\max_a Q_k(a)$。

在选择行为 a 之后接收奖励值 $r(a)$，并更新行为 a 的 Q 值，Q 值更新公式如下：

$$Q_{k+1}(a) = Q_k(a) + \beta_k[r_{k+1}(a) - Q_k(a)] \quad (1)$$

其中：$r_{k+1}(a)$ 为行为 a 在 k 时刻的奖励；$\beta_k (0 < \beta_k \le 1)$ 为一个步长大小参数。

固定步长 β 适用于静态过程。当 β 接近 1 时，可获最近的奖励；当 β 接近 0 时，可获得先前的奖励。一般将静态网络的 β 设为 0.1。静态网络中，β_k 若满足如下条件，则可以确保收敛到概率为 1。

$$\sum_{k=1}^{\infty} \beta_k = \infty \quad 且 \sum_{k=1}^{\infty} \beta_k^2 < \infty \quad (2)$$

假设 $\beta_k = 1/(k+1)$ 满足条件式(2)，且可以得到过去奖励的平均值。当 $\beta_k = \beta$ 时，估计过程将不会完全收敛，会根据最新观测的奖励值而变化。$\beta_k = \beta$ 时，所形成的奖励的加

权平均为

$$Q_{k+1}(a) = (1-\beta)^{k+1} Q_0(a) + \sum_{i=1}^{k+1} \beta(1-\beta)^{k+1} r_i(a) \quad (3)$$

式(1)所表述的传统 ε -贪婪学习策略中由于固定 β 参数，致使其易陷入局部最优。而式(3)所表述的改进型学习策略中，设置 $\beta = 1/(k+1)$，使其能够根据学习进程适时调整比例参数，使算法能够跳出局部最优，直到获得全局最优为止。为此，本文采用了这种改进型的 ε -贪婪学习策略。

3　提出的网页排序算法：GLUB-Rank

本文提出了一种基于强化学习和用户反馈的排序算法，利用了查询—网页对的数据特征和列表法。

该算法的思想是对查询—网页对的特征进行评估，每个特征代表网页或查询的某个方向，并使用多个特征来涵盖缺失。换句话说，与用户观点相关的页面可以具有与其他相关页面类似的内容或链接。然后，考虑多个相关特征来提供额外的结果，这就减少了单独使用每个特征的缺点。

3.1　网页排序基本步骤

环境包含了用户和网页。作为代理的排序系统会选择 10 个网页，并将其显示给用户。根据用户的点击行为，网页特征的重要程度会根据特定策略进行奖励或惩罚。在此过程中，每次迭代都会将网页排序到一个用户的推荐列表中。

排序的第一阶段首先从网页中选择 10 个网页并向用户显示，这种网页的选择是利用贪婪方法和旋转轮策略进行。首先，在列表的任何位置都有选择网页的概率，而且随着时间的推移选择前 10 个网页的概率会增加。因为在此时，相关网页已移动到列表顶部，以此提高获得相关网页的有效性和减少对环境的搜索。

第二步是根据网页的重要程度（行为值）确定网页情况，即根据行为值来选择网页。

然后，用户点击其中一些网页。检查被点击的网页，如果网页是相关且网页特征反映了相关性，那么重要程度会相应的增加；如果网页是不相关的且网页特征也显示不相关性，那么重要程度会相应的减少。

如果网页是相关的，那么与网页—查询对相关的用户反馈特征会被奖励，否则将被惩罚。在每个步骤中，当用户反馈是其所看到的最后一个网页时，网页特征的重要程度不会被奖励或惩罚。当时间达到截止时间或用户查询失效时，该重复过程结束。

值得注意的是，在网页排序中，如果有更多的网页特征，则可以更好地描述网页，进而有助于区分相关和不相关的网页。

3.2　用户反馈

用户点击网页是用户反馈的主要方式。用户的点击行为会受到干扰，即用户时常不能准确地点击相关网页，仍然会点击最高级别的无关网页。这表明即使列表中的第一级网页与查询不相关，但这些网页的点击频率还是很高。

在这种情况下，根据所获得的频率，考虑了一种包含对列表顶部 10 个网页进行点击的 11 种情况和 10 种概率的统计模型。

3.3　结合 ε -贪婪学习和用户反馈的网页排序

在提出的 GLUB-Rank 算法中，假定强化信号是恒定的，并且根据相同的特征和相关程度，在分类网页列表中根据所点击网页的位置来接收强化信号。同时也考虑了一些其他特征来确定特征的重要程度。这些特征更好地描述了网页，提升了网页重要程度评估的准确性。

GLUB-Rank 算法步骤如下：首先，用户输入查询，排序系统作为代理向用户显示网页列表。每个列表根据网页特征进行排列，并且在学习过程中，这些网页特征的位置是不会改变的。用户点击网页，奖励或惩罚机制对点击网页的特征进行奖励。排序系统通过操作选择要显示的网页，并向用户显示主要网页列表。

主列表中的网页是根据优先级，使用 ε-贪婪方法所选择的。随着时间的推移，选择前 10 个网页的概率随着 ε 的增加而增加。起初由于反馈知识匮乏，ε 的值为零，概率选择类似于轮盘赌操作，根据列表中网页优先级确定网页位置。随着用户对环境的了解越来越多，相关网页会排列在最高等级上，并将具有更高的优先级。另一方面，随着时间的推移，根据从环境中获取的知识，使得下层网页被认为是不相关的网页，因此，这些网页的被选择的概率接近于零。网页选择的概率根据以下公式计算：

$$m(x) = \begin{cases} \dfrac{2*(\beta*(n+1-x)-(v*t))}{n*(\beta*(n+1)-2*v*t)}, & 1 \le x \le [n - \dfrac{v*t}{\beta}] \\ 0\ , otherwise \end{cases} \quad (4)$$

$$p(x) = \begin{cases} 0.5*(m(x)+\varepsilon+\dfrac{1-\varepsilon}{n}), (1 \le x \le 10 \quad 0 \le \varepsilon \le 0.1) \\ 0.5*(m(x)+\dfrac{1-\varepsilon}{n}), (10 \le x \le n \quad 0 \le \varepsilon \le 0.1) \end{cases} \quad (5)$$

其中：$m(x)$ 表示增量轮盘方法中的概率，随着时间的推移，网页处于低级别的概率变为零；x 为网页在有序网页列表中显示的位置；n 为每个相关查询的网页数量；v 的初始值等于 1 且线性增长；t 是时间。每一轮中，在用户查看所有查询结果后，t 的值会增加。

重复这个过程，直到主列表收敛为一个固定的列表，训练阶段结束。每个特征的重要程度是通过将特征值的总和乘其权重，表示每个查询-网页对测试阶段的得分。

训练阶段中，对每个网页考虑 k 个特征，目标是将基于特征和用户反馈的排序列表收敛到一个固定列表。在此过程中需

要确定特征的重要程度，表示为 E 。开始时，重要程度被认为是相同的($E=1/k$)，并将每个列表中的网页按照相应特征的值排序。同时标记网页的点击位置，以对相应特征进行评分。然后，为查询的相关网页建立主列表，通过式（6）计算网页的分数，并以降序排列向用户显示。

$$Score_{d,q}=\sum_{i=1}^{k}E_i*F_{i_{d,q}}+k*fe_{d,q} \qquad (6)$$

其中：$Score_{d,q}$ 表示查询-网页对的得分；k 是特征数量；$F_{i_{d,q}}$ 是与查询—网页对相关的第 i 个特征的值；E_i 表示第 i 个特征对应的重要程度；$fe_{d,q}$ 表示用户对所配对的查询—网页 $(d-q)$ 反馈的特征值。

在网页排序中，根据优先级随机方法（轮盘和 ε-贪婪的组合）选择 10 个向用户显示，用户将点击显示网页。如果相关，则用户反馈的特征将会得到奖励，否则就会受到惩罚。换句话说，如果该特征对网页正确排序有贡献，则给予奖励，否则将被处罚。重要程度的计算如下：

$$\alpha=e^{-\beta t} \qquad (7)$$

$$e_i(t+1)=e_i(t)+\alpha\times\left[-e_i(t)\mp r\right] \qquad (8)$$

$$fe_{d,q}(t+1)=fe_{d,q}(t)+\alpha\times\left[-fe_{d,q}(t)\mp r\right] \qquad (9)$$

其中：α 表示学习率；β 表示步长，数值为 0.01；t 是时间。$t=0$ 时的学习率为 1，随着时间的推移接近零，学习完成；r 表示奖励，这个参数的值是不变的；$fe_{d,q}(t)$ 表示在 t 时刻查询-网页对的用户反馈值；$e_i(t)$ 表示在 t 时刻对应于第 i 个特征的权重。重复学习过程，使网页列表收敛到一个固定的列表。

提出的 GLUB-Rank 算法的伪代码如算法 1 所示，流程如图 2 所示。

算法 1：GLUB-Rank 算法

输入

d, //查询-网页矩阵

q, //查询列表

value_click，//用户点击 web 搜索结果的相关概率分布，为 10 * 11 维矩阵

v，//网页特征值矩阵。

输出

E, //特征的重要程度

Rank_test，//最终排名列表。

假设

if $\exists(q_i,d_j)$ **then** $\exists d_{i,j}$;

初始化

t= 0,punish = -1*reward, *β*=0.01；

for all vf **Set** $vf_{ij}=0$；

for *i*=1,2,…,*g* **repeat Set** *E*=1/*g*;

for *i*=1,2,…,*g* **repeat Set** 一个与使用第 *i* 特征查询相关的网页排序列表。

开始

while t<N

　$\alpha=e^{-\beta t}$； //学习率 0-1 之间

　for *i*=1 to *n*

　　通过 $vf_{ij}*g+sum(E_i*v_{(d,q)_i})$ 来选择 10 个网页构成排序列表网页 R；

　　将排序列表 R 显示给用户；

　　PTR =列表 R 中第一个相关网页($d_{i,h}$) 的位置；

　　用户根据概率 $value_click_{PTR,i}$ 点击列表 R 中的网页；

　　if 网页是相关的 **then** //奖励

　　　for *p* = 1 to *g* //对所有特征

　　　　if 该网页在第 *p* 个特征排序列表的顶端位置 **then**

　　　　　$e_p(t+1)=e_p(t)+\alpha\times\left[reward-e_p(t)\right]$；

　　　　else

　　　　　$e_p(t+1)=e_p(t)+\alpha\times\left[punish-e_p(t)\right]$；

　　　　end

　　　end

　　　$vf_j(t+1)=vf_j(t)+\alpha\times\left[reward-vf_j(t)\right]$；

　　else if 网页是不相关的 **then** //惩罚

　　　for *p*=1 to *g* //对所有特征

　　　　if 该网页在第 *p* 个特征排序列表的底部位置 **then**

　　　　　$e_p(t+1)=e_p(t)+\alpha\times\left[reward-e_p(t)\right]$；

　　　　else

　　　　　$e_p(t+1)=e_p(t)+\alpha\times\left[punish-e_p(t)\right]$；

　　　　end

　　　end

　　　$vf_j(t+1)=vf_j(t)+\alpha\times\left[punish-vf_j(t)\right]$；

　end

　t=*t*+1；//直到查询会话结束

```
end
for i=1 to m        //所有网页
    For j=1 to n  //所有查询
        rank_test_{i,j} = Σ v_{(i,j,k)} * e_k ;
    end
end
```

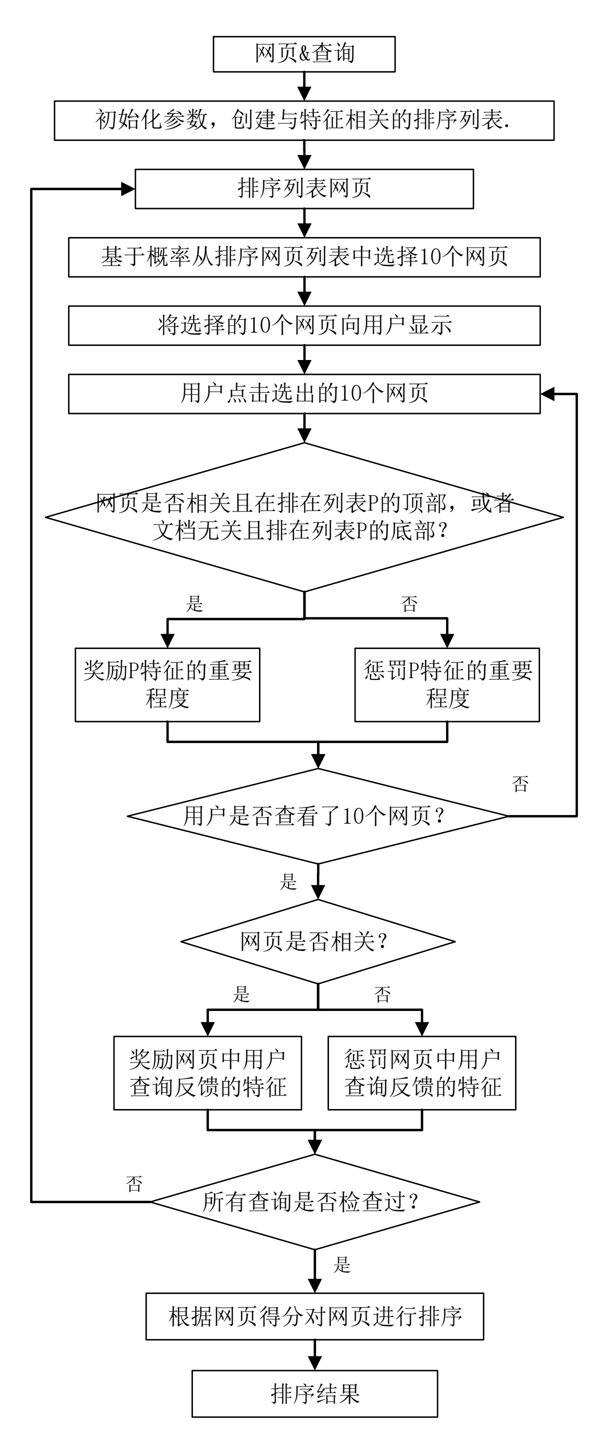


图 2　GLUB-Rank 算法的流程

4　实验及分析

在相同的条件下对提出的方法进行评估和比较。与排序相关的基准数据集有网页集、查询格式信息以及与查询—网页对相关的人为判断三个组成部分。

对于本文方法，设置学习轮次为 100，奖励值在[0.5,0.85]内，考虑的特征数目为 17。

4.1　评估标准

广泛应用于评估信息检索的评估标准有前 n 位的精确度（P@n)、平均精确度（MAP）和归一化折损累积增益（NDCG）。分别描述如下：

$P@n$ 标准显示了在每个查询对应的最终排序列表中，排名前 n 个网页中是相关网页的数量，即取前 n 个查询结果，计算其查准率，公式如下：

$$P@n = \frac{NoR_n}{n} \tag{10}$$

其中：NoR_n 表示相关网页排在前 n 位的数量。

MAP 表示为所有查询准确率(AP)的平均值，相关网页在系统推荐列表中越靠前，MAP 就越高。对于每个查询的 AP，其表示为相关网页 $P@n$ 的平均值，表示如下：

$$AP = \frac{\sum_{n=1}^{N}(P@n \times R_e(n))}{T_R} \tag{11}$$

其中：N、T_R 和 $R_e(n)$ 分别代表检索到的网页数量、相关网页数量和第 n 个相关网页的二进制函数，函数值为 1 表示相关网页，为 0 表示无关网页。

DCG 是折损增益值，等级高的网页在推荐列表中的位置越靠前，则该值越高。$NDCG@n$ 是对 DCG 进行归一化，表示如下：

$$NDCG@n = Z_n \sum_{m=1}^{n} \frac{2^{r(m)}-1}{\log(m+1)} \tag{12}$$

其中：$r(m)$ 表示排序列表中第 m 个网页的相关率；Z_n 是归一化常数；$2^{r(m)}-1$ 表示第 m 个网页的增益；$\frac{2^{r(m)}-1}{\log(m+1)}$ 表示增益折损；$\sum_{m=1}^{n}\frac{2^{r(m)}-1}{\log(m+1)}$ 表示第 n 个位置的累计增益折损。

4.2　基准数据集

本文应用了 LETOR3 版本的 OHSUMED 数据集。OHSUMED 是一个医学期刊网络数据库 MEDLINE 的一个子集，该集合包含了 348 566 条记录。将每篇文献网络页面作为一个网页。另外，该集合由 106 个查询组成，16 140 个具有相关度的查询—网页对。OHSUMED 包含 45 个特征，这些特征由查询—网页对决定，其中一些特征不依赖于查询—网页对。另外，用户判断类型有三个：相关的、部分相关的和不相关的。

4.3　性能评估

将本文提出的 GLUB-Rank 方法与 RankBoost 和 RankSVM

算法进行比较。其中，设置推荐列表的长度 n=5、10 和 15 三种情况。

图 3、4 显示了当 n=10 时，各种算法在 P@10 和 NDCG@10 方面的性能比较。其中横坐标表示前 10 个列表中的位置。表 1 给出了 n=5、10 和 15 时，各种算法的平均 P@n、NDCG@n 和 MAP@n 值。

通过分析可以看出，各种方法在排序列表的前面位置上都具有较好的性能，位置越后性能越低。另外，在不同的 n 值下，n 值越大，各种性能指标也有所下降。这是因为一般排序算法对相关性前几名的相关网页排序的准确性都较高，越到后面正确排序的难度越大。

从各种方法的比较来看，本文 GLUB-Rank 方法具有优越的性能。虽然本文采用的列表法和其他方法采用的成对法中都利用了用户反馈和列表特征，但是列表法呈现出更好的结果。对于 RankSVM 方法，本文采用的在线强化学习的性能比 SVM 分类器更好。另外，侧重于中低级别网页是 RankSVM 方法中存在的问题之一，其最终排序模型会受到更多相关性网页查询的强烈影响。相比之下，GLUB-Rank 算法侧重于找到相关网页并将其插入到最高层次。同时，由于 GLUB-Rank 算法从特征中取值，所以不管是否要查询都需要计算这些值。因此，该算法不会有 RankSVM 问题。相比而言 RankBoost 算法的性能优于其他两种对比方法，这证明了基于强化学习能够提高性能。

总的来说，提出的 GLUB-Rank 算法能够给出合理的网页排序，在 P@n 和 NDCG 评价标准方面具有很大的优越性，在 MAP 方面也与现有的较佳方法保持一致。

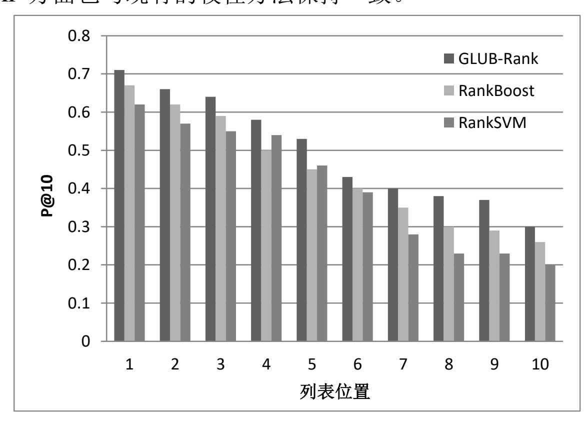


图 3　各种算法的 P@10 性能比较

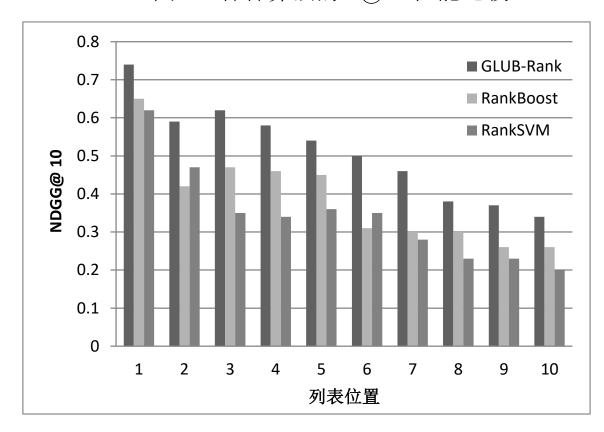


图 4　各种算法的 NDCG@10 性能比较

表 1　不同 n 值下各种算法的平均性能比较

性能指标	列表长度	排序算法		
		GLUB-Rank	RankBoost	RankSVM
P@n	n=5	0.531	0.479	0.431
	n=10	0.503	0.443	0.407
	n=15	0.486	0.428	0.382
NDCG@n	n=5	0.543	0.406	0.359
	n=10	0.512	0.388	0.343
	n=15	0.495	0.367	0.323
MAP@n	n=5	0.452	0.431	0.408
	n=10	0.447	0.425	0.403
	n=15	0.443	0.420	0.397

对于运行时间，RankSVM 算法的训练和推荐执行时间最长，所有样本训练完大约需要 55 min，平均每次查询的推荐时间为 1.3 s。这与其所选择的核函数有关。另外 SVM 中通过交叉验证来选择模型最优参数也耗时较大。RankBoost 算法的训练时间较短，训练时间约为 34 min，这说明 Boost 强学习排序器学习速度较快。在不考虑用户反馈停留时间下，本文 GLUB-Rank 算法的学习时间略大于 RankBoost 算法，约为 40 min，查询推荐时间约为 1.1 s。虽然本文方法速度不是最快，但推荐精度是三者中最优越的。

5　结束语

本文利用一种强化学习方法来构建网页推荐系统，排序系统将用户反馈作为强化信号，减少了固有干扰对用户点击的影响。首先根据用户查询向用户推荐初始网页列表；然后根据用户点击网页的行为反馈进行ε-贪婪学习，获得每个网页的相关性程度值，并以此对网络排序进行调整优化。在 OHSUMED 数据集上的实验结果表明，提出的算法可以获得较高的推荐准确性。

在今后研究中，将考虑不同的特征组合和特征数目来提高算法的性能。

参考文献：

[1] 李又玲，常致全，杨浩. 多标签网页的 Gauss-PNN 粗糙集排序推荐 [J]. 计算机应用研究, 2017, 34 (2): 382-385.

[2] 李兰英，周秋丽，孔银，等. 子图估算 PageRank 网页排序算法研究 [J]. 哈尔滨理工大学学报, 2017, 22 (2): 117-123.

[3] Singhal R, Srivastava S R. Enhancing the page ranking for search engine optimization based on weightage of in-linked web pages [C]// Proc of International Conference on Recent Advances and Innovations in Engineering. Piscataway, NJ: IEEE Press, 2017: 1-5.

[4] 陈松乐，孙正兴，张岩，等. 一种运动数据检索的相关反馈算法 [J].

电子学报, 2016, 44 (4): 868-872.

[5] Joachims T, Granka L, Pan B, *et al.* Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search [J]. ACM Trans on Information Systems, 2017, 25 (2): 7-16.

[6] 曹姗姗, 王冲. 基于网页链接与用户反馈的 PageRank 算法改进研究 [J]. 计算机科学, 2014, 41 (12): 179-182.

[7] Jin X B, Zhang D, Yu J, *et al.* Multi-partite ranking with multi-class AdaBoost algorithm [C]// Proc of International Conference on Fuzzy Systems and Knowledge Discovery. Piscataway, NJ: IEEE Press, 2012: 884-887.

[8] Song Y, Leung K, Fang Q, *et al.* FP-Rank: an effective ranking approach based on frequent pattern analysis [C]// Proc of International Conference on Database Systems for Advanced Applications. Berlin: Springer, 2013: 354-369.

[9] Bidoki A M Z, Yazdani N. DistanceRank: an intelligent ranking algorithm for web pages [J]. Information Processing & Management, 2008, 44 (2): 877-892.

[10] Ghansah B, Wu S, Ghansah N. Rankboost-based result merging [C]// Proc of IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. Piscataway, NJ: IEEE Press, 2015: 907-914.

[11] 聂慧, 彭娇, 金晶, 等. 基于多核系统的并行线性 RankSVM 算法 [J]. 计算机应用研究, 2017, 34 (1): 46-51.

[12] 谭咏梅, 王睿, 李茂林. 基于上下文信息和排序学习的实体链接方法 [J]. 北京邮电大学学报, 2015, 38 (5): 33-36.

[13] Sawant U, Chakrabarti S. Features and aggregators for Web-scale entity search [J]. Eprint Arxiv, 2013, 24 (4): 13-19.

[14] Oosterhuis H, De Rijke M. Balancing speed and quality in online learning to rank for information retrieval [C]// Proc of ACM on Conference on Information and Knowledge Management. New York: ACM Press, 2017: 277-286.

[15] Torkestani J A. An adaptive learning to rank algorithm: learning automata approach [J]. Decision Support Systems, 2012, 54 (1): 574-583.